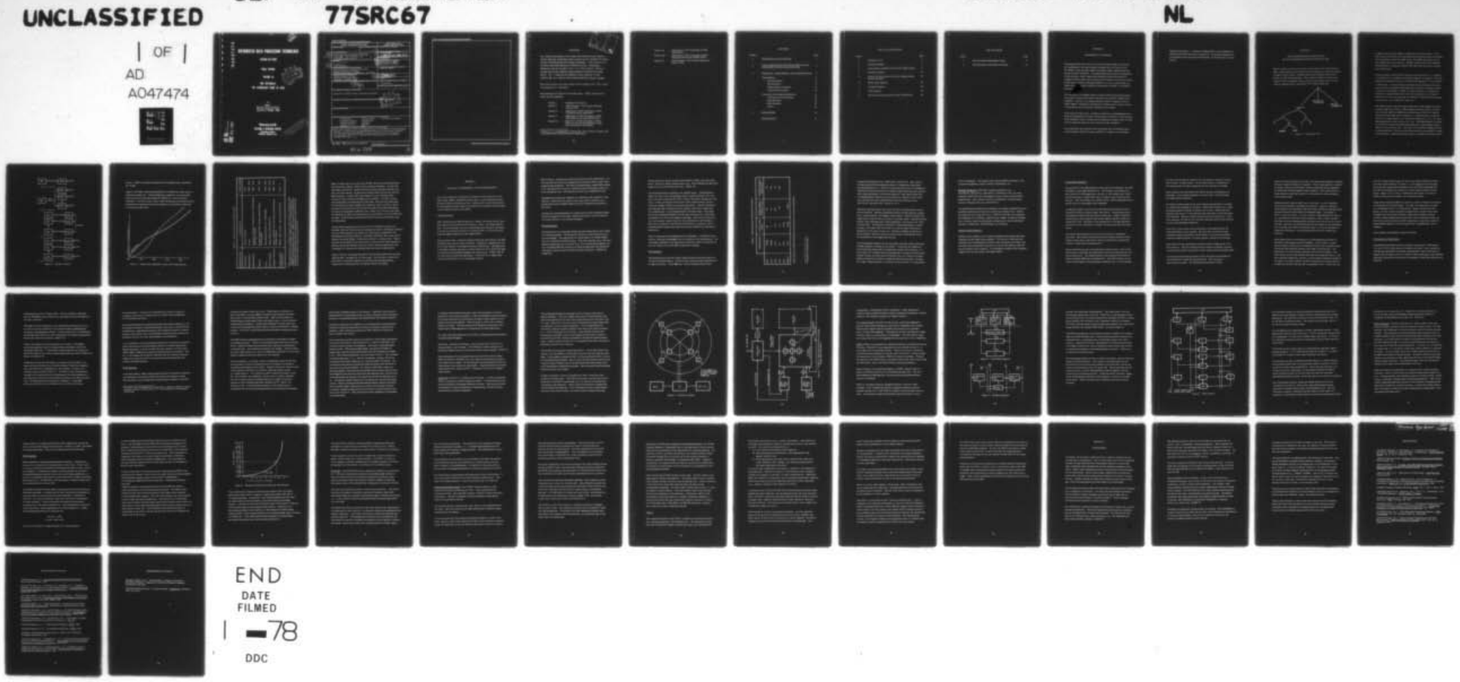
AD-A047 474 HONEYWELL INC MINNEAPOLIS MINN SYSTEMS AND RESEARCH --ETC F/G 9/2
DISTRIBUTED DATA PROCESSING TECHNOLOGY. VOLUME III. DDP RATIONA--ETC(U)
SEP 77 R RAMSEYER DASG60-76-C-0087

UNCLASSIFIED 77SRC67 NL

| OF |
AD
A047474



END
DATE
FILMED
1 -78
DDC

DISTRIBUTED DATA PROCESSING TECHNOLOGY

DASG60-76-C-0087

FINAL REPORT

VOLUME III

DDP RATIONALE:

THE TECHNOLOGY POINT OF VIEW

For

BMDATC
Ballistic Missile Defense
Advanced Technology Center
Huntsville, Alabama 35807

Honeywell
SYSTEMS & RESEARCH CENTER
2600 RIDGWAY PARKWAY
MINNEAPOLIS, MINNESOTA 55413

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOV'T ACCESSION NUMBER

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (AND SUBTITLE)

6 Distributed Data Processing Technology. Volume III. DDP Rationale: The Technology Point of View.

5. TYPE OF REPORT/PERIOD COVERED

9 Final Rep[redacted]t.
Oct[redacted] [redacted]76 ~~to~~ Oct[redacted] [redacted]77,

6. PERFORMING ORG. REPORT NUMBER

14 77SRC67

7. AUTHOR(S)

10 R. Ramseyer

8. CONTRACT OR GRANT NUMBER(S)

15 DASG60-76-C-0087

9. PERFORMING ORGANIZATIONS NAME/ADDRESS

Honeywell Systems and Research Center
Computer Systems Technology
2600 Ridgway Parkway, Minneapolis, MN 55413

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME/ADDRESS

Ballistic Missile Defense
Advanced Technology Center
Huntsville, Alabama 35807

12. REPORT DATE

11 Sep[redacted] [redacted]77

13. NUMBER OF PAGES

56

14. MONITORING AGENCY NAME/ADDRESS (IF DIFFERENT FROM CONT. OFF.)

12 55p.

15. SECURITY CLASSIFICATION (OF THIS REPORT)

Unclassified

15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (OF THIS REPORT)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (OF THE ABSTRACT ENTERED IN BLOCK 20, IF DIFFERENT FROM REPORT)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

Architecture	Microprocessor	Distributed Processing
Distributed System	Uniprocessor	
Parallel Processor	Emulation	
Array Processor	Network	
Pipeline	LSI	

20. ABSTRACT (CONTINUE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

This report examines some of the salient points in the evaluation of data processing systems. An overview of the classification scheme for architectures is followed by somewhat detailed examinations of classic examples of each architecture. The advantages and future of distributed processing systems are pointed out throughout the report.

HD-168 REV 11/74

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 55 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)

402 349

LB

FOREWORD

The research documented in this volume was conducted under Ballistic Missile Advanced Technology Center contract number DASG60-76-C-0087, entitled "Distributed Data Processing Technology." This work was performed by Honeywell Systems and Research Center, Minneapolis, Minnesota under the direction of Mr. C. R. Vick, Director, Data Processing Directorate, Ballistic Missile Defense Advanced Technology Center. Mr. J. Scalf was the BMDATC project engineer for this contract; Ms. B. C. Stewart was the Honeywell/GRC program manager.

This report covers work from October 1976 to October 1977. This volume was prepared by R. Ramseyer.

This document is Volume III of the final report. Other volumes of the report are the following:*

Volume I	-	Management Summary
Volume II	-	DDP Rationale: The Program Planning Point of View
Volume IV	-	Application of DDP Technology to BMD: Architectures and Algorithms
Volume V	-	Application of DDP Technology to BMD: DDP Subsystem Design Requirements
Volume VI	-	Application of DDP Technology to BMD: Impact on Current DP Subsystem Design and Development Technologies

---

*Volumes V, VI, the appendix to Volume VII, and a section of Volume VIII were prepared by General Research Corporation.

Volume VII	-	Application of DDP Technology to BMD: Experiments
Volume VIII	-	Application of DDP Technology to BMD: Research Performance Measurement
Volume IX	-	DDP Rationale: The Program Experience Point of View

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

BACKGROUND & OVERVIEW

Components which were only a very short time ago thought to be beyond the capabilities of the semiconductor manufacturers are almost daily becoming a reality through rapidly developing, highly sophisticated integrated circuitry techniques. The exponential-like increase of capabilities in LSI and soon, VLSI, coupled with parallel evolution/maturation of computer science itself, has created and will continue to create new forms of data and/or signal processing equipment at an ever accelerating rate. These changes continue to influence the potential "payoffs" of distributed processing.

The complexity and perhaps novelty of many of these processing systems is beyond the scope of this report and often may preclude categorization altogether. However, an attempt should be made to categorize and to a lesser degree, catalogue at least generic architectures in order to obtain a firmer grasp of the meaning of distributed processing.

In the context of this report, computer architecture is distinguished from hardware implementation detail, and therefore is defined as "the structure of the computer which a machine-level programmer needs to know in order to write programs which will run correctly on that computer."

The architecture thus consists of the instruction set, the address space, and the registers which are made available or "visible" to the machine

language programmer. In general, nothing which is time-dependent is considered (strictly) part of an architecture. Comparing computers is much different than comparing architectures, as will become very clear shortly.

SECTION 2

THE CLASSIFICATION AND EVOLUTION OF DATA PROCESSING SYSTEM ARCHITECTURES

Figure 1 shows a tree structure which can be used to depict the decision space within which one can classify most architectures. The first decision is single control stream, multiple control stream (MCS) with no interaction, or MCS with interaction. The concept of stream is used in this report to mean a sequence of items (data or instructions) which are executed or operated on by a processor. [FLYN 66], [FLYN 72]

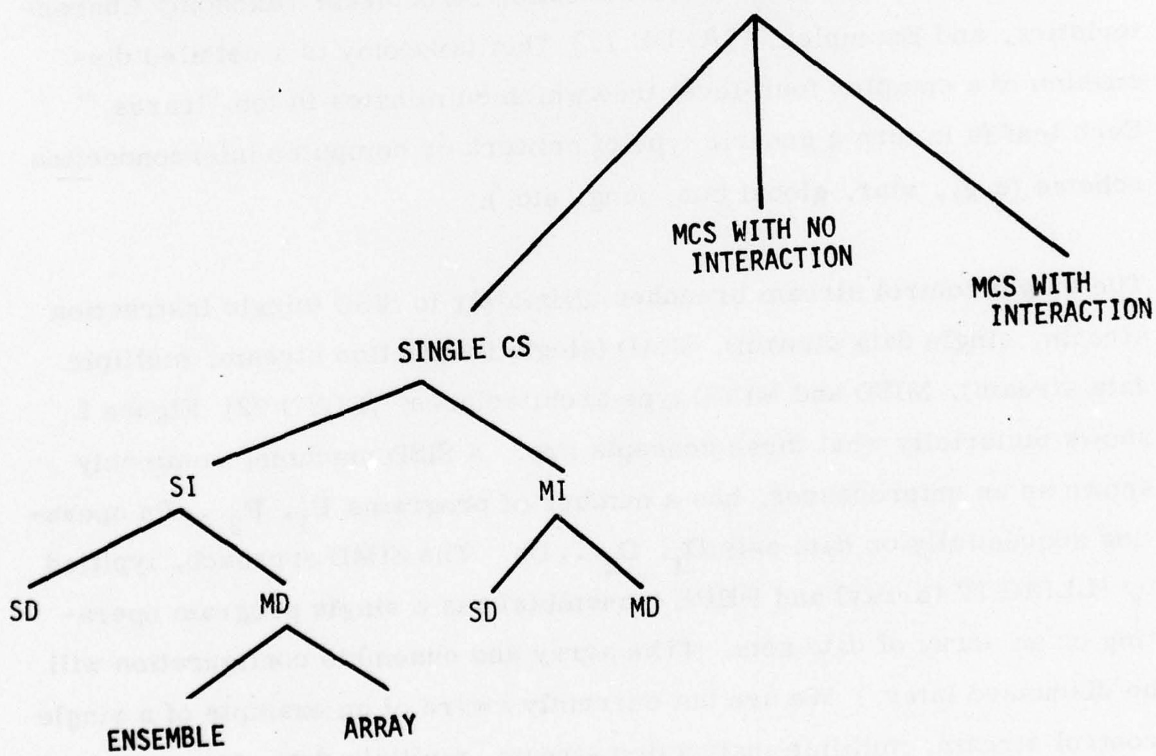
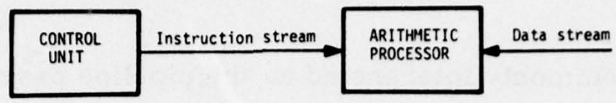


Figure 1. Taxonomic Tree

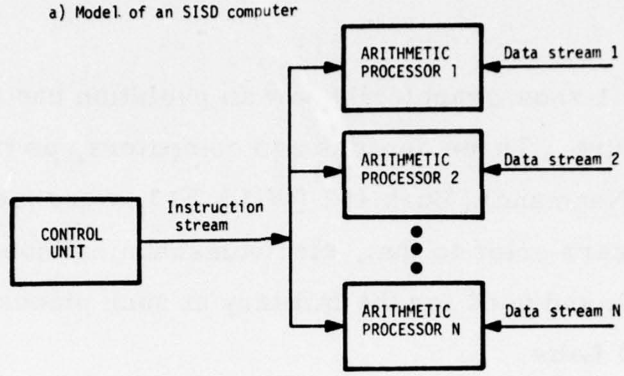
A single control stream implies a single source of instructions. At the instruction stream level, the instruction may cause more than one machine-level instruction to occur (in other words, the single instruction stream and multiple instruction stream). Multiple control stream then suggests multiprocessors or special machines with more than one source for high-level instructions.

If these multiple control streams interact with each other (i.e., cooperate or communicate), the right-most branch of Figure 1 is invoked. The most complex branching in the tree, at the MCS with interaction, is not covered in detail here. A more complete treatment of these branches and leaves may be found in "Computer Interconnection Structures: Taxonomy Characteristics, and Examples." [ANDE 75] This taxonomy is a detailed discussion of a complex four-level tree which culminates in ten "leaves." Each leaf is in turn a generic type of network or computer interconnection scheme (e.g., star, global bus, ring, etc.).

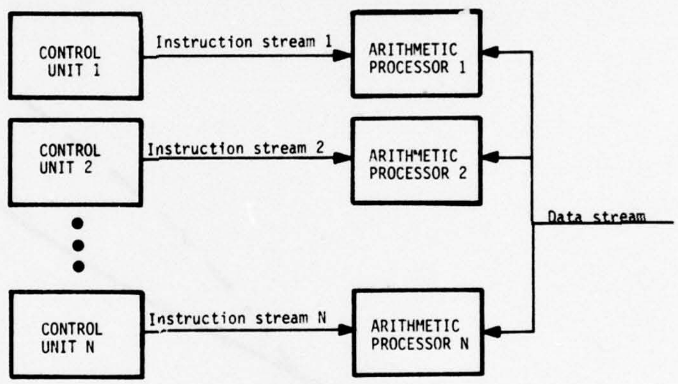
The single control stream branches ultimately to SISD (single instruction stream, single data stream), SIMD (single instruction stream, multiple data stream), MISD and MIMD type architectures. [FLYN 72] Figure 2 shows pictorially what these concepts are. A SISD machine, commonly known as an uniprocessor, has a number of programs P_1, P_2... Pn operating sequentially on data sets D_1, D_2... Dn. The SIMD approach, typified by ILLIAC IV (array) and PEPE (ensemble) has a single program operating on an array of data sets. (The array and ensemble configuration will be discussed later.) We are not currently aware of an example of a single control stream, multiple instruction stream, multiple data stream archi-



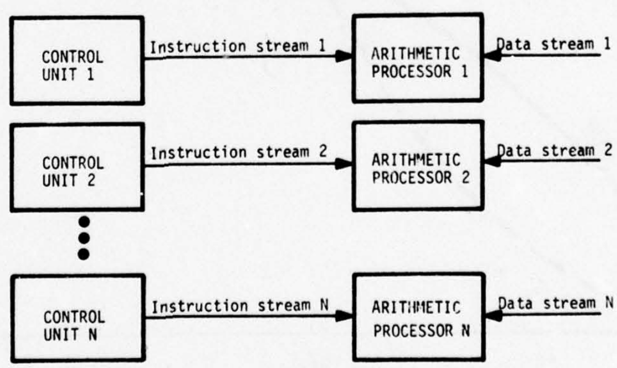
a) Model of an SISD computer



b) Model of an SIMD computer



c) Model of an MISD computer



d) Model of an MIMD computer

Figure 2. Computer Models

tecture. MISD is commonly interpreted as the pipeline case, typified by the TI-ASC.

Figure 3 and Table 1 show graphically how an evolution has taken place in computer architecture. Three decades ago computers, as most people think of them (von Neumann) [Burk 46] [WILK 51], were just being developed. Five years prior to that, electromechanical nonstored program machines were built and used for the military at such places as Harvard University and Bell Labs.

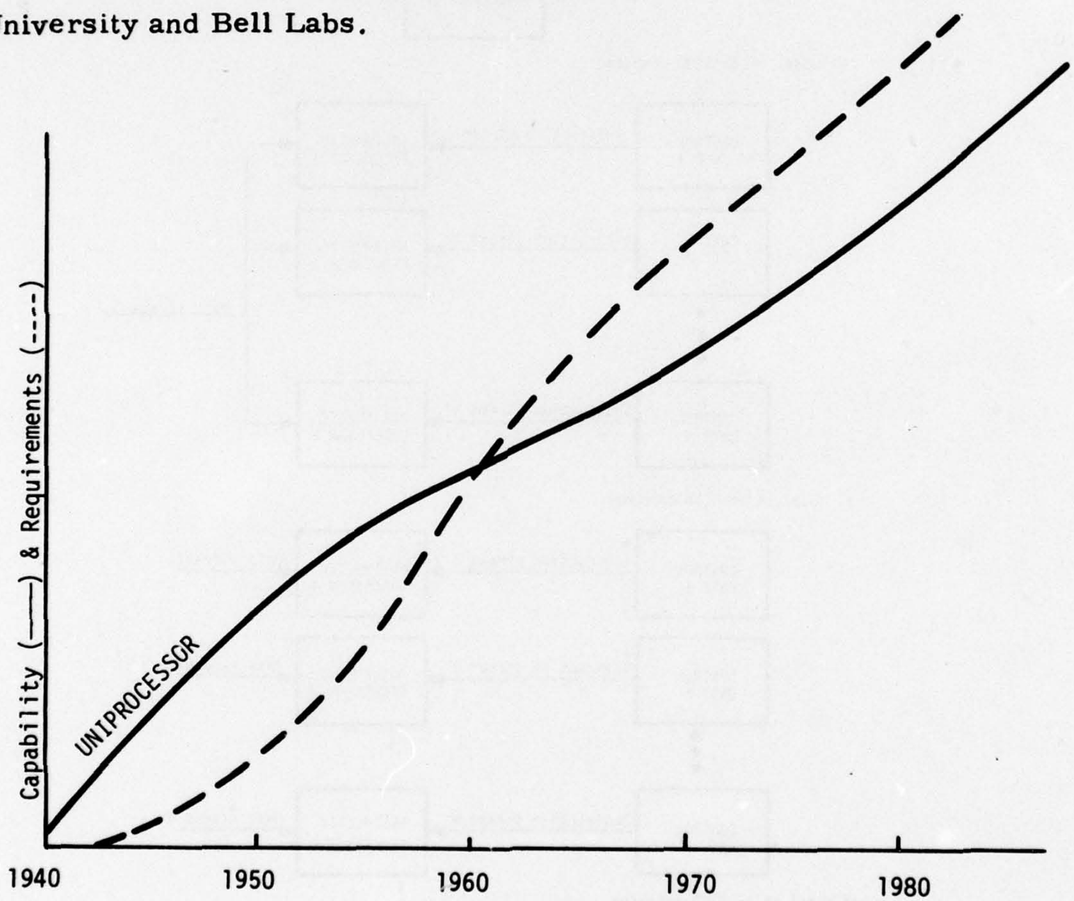


Figure 3. Uniprocessor Capability versus User Requirements

Table 1. Data Processing Technology Trends

ARCHITECTURE	APPLICATION	TIME FRAME
von Neumann	Bureau of Census	1951
Multicomputer	Stretch	1962 - 1972
Multiprocessor	Airline Reservations, Safeguard CLC, SITE Defense CDC 7700, Arts III	1962 - 1974
Pipeline	Weather Bureau/ASC, Weapons Research/Star, FFA/STARAN	1970 - 1972
Array	DOD Research/ILLIAC IV, PEPE/BMD	1970 - 1976
Special Purpose	Digital Signal Processing/FFT	1968 - 1977
Computer Networks	ARPA, TYMNET (Time Sharing) Process Control, Pipeline Control	1965 - 1977
Problem-Driven* ("Distributed")	MMBC, Point-of-Sale, Smart Instrument Nets	1977 - ?

*This may consist of a combination of any or all previous architecture types integrated at one or more levels of abstraction for a specific application problem and its associated objectives. BMD's need is for problem-driven architectures (PDAs) rather than for distributed processing architectures (in the usual computer science sense of the term).

Figure 3 shows that in the early days of EDP, the data processing systems were more than capable, albeit crude by modern standards. As time and technology marched on, computers with increased capability became available, but unfortunately, computational loading caught up quite fast with the SISD machines. Around 1960, users began to turn to multiprocessor systems for their required throughput and reliability. By 1964, Control Data Corporation was delivering the CDC 6600, which provided pipelining of instructions and used multiple (ten) arithmetic units to achieve a speed increase of more than two and one-half times that of the CDC 6400 (which had only one comparable arithmetic unit). [BELL 71] The first CDC 7600 was delivered in 1969, and eventually in the early 1970's, the CDC STAR 100 emerged. Each machine delivered marked increases in performance over its predecessor.

Control Data Corporation was not alone in the evolutionary period, but the 6600, 7600 and STAR are prime examples as well as well-known machines. The late 1960's produced the ILLIAC IV, which was an outgrowth of the SOLOMON work done by Slotnick. [SLOT67], [SLOT62] More recent developments have been PEPE, TI ASC and the CRAY-I, to name a few. These systems are motivated by two main forces: 1) the need for large systems to perform special processing, and 2) the need to take advantage of new LSI technology as it emerges by utilizing repetitive system organizations.

Figure 3 shows a continuing divergence between SISD performance and user requirements, although both are increasing. The only way to narrow the gap will be a continued emphasis on distributed processing systems, coupled with continued growth in integrated circuit technology.

SECTION 3

EXAMPLES, COMPARISONS, AND CONSIDERATIONS

This section discusses and shows by example the characteristics of the SISD, SIMD, MISD, and MIMD machine types. It also discusses the differences within an architecture (e.g., speed, power, cost) and discusses the circumstances under which one architecture or specific machine within a class might be chosen over another.

UNIPROCESSORS

When considering the SISD machines as a family, one finds a nearly complete spectrum of performance parameters, prices, and simple physical size. Strictly speaking, the term uniprocessor thus applies to something as small and relatively primitive as an INTEL 4004, as well as the very large and highly sophisticated Burroughs B7800.

When removed from a relatively simple batch mode environment, the SISD machines have several severe drawbacks. In general, uniprocessors are not as fault-tolerant as multiple processor architectures. Although some of the large machines may have a certain amount of error correction logic in memory and distributed throughout the system (e.g., B7700), they do not truly allow for graceful degradation. If the CPU (or a single main memory module) fails, it is a catastrophic failure.

SISD machines, although often used for real-time control applications, are far from ideal for this type of application and usually require tricky multi-programming techniques. The costs of programming a single SISD machine to do a job equivalent to that of a relatively simple distributed system become outrageous (e.g., the AN/UYK-16 used in the AN/SQS-56 sonar).

Expandability is generally relegated to a simple but often expensive trade upwards. Unless the user sticks to the same vendor, however, he is usually faced with incredible software and I/O compatibility problems in this trade.

The general comprehensibility of a uniprocessor and the relatively simple executive program are the major reasons why there is and will continue to be much inertia to stay with the monoliths.

Microprocessors

Microprocessors are presently divided into three generations, all of which are currently popular. The generation division is made largely by hardware technology. First generation microprocessors are usually PMOS, four-bit, calculator-oriented devices. They have typically up to a 4K-word address capacity and are very slow (e.g., 62.5 microsecond register add for a Fairchild PPS-25), but they are cheap and generally well-suited for such uses as point-of-sale terminals or hand calculators. [OGDE 74], [CUSH 75]

Second generation micros are generally NMOS or CMOS, 8-bit word size, faster, and have a larger address space (e.g., RCA COSMAC has 64K-word range and a six-microsecond add time). [SWAL 74]

The third generation is typified by the AM2900 family. This generation differs not only in the IC technology which is typically bipolar (TTL, IIL, ECL), but also in that they have been "sliced" vertically, usually into 4-bit slices. These processors are fast due to their bipolar nature, but are also power-hungry relative to previous generations. This has resulted in less logic per chip, which in turn has resulted in the slice concept and, more significantly, a microprogrammable microprocessor. One might say the microprogrammability of the bit slice processors is a fortuitous consequence of a flaw in the bipolar technology. Since there is less hardware on the chips, control cannot be very complicated, and therefore, a good deal of emulation of the control via firmware is called for. Microprogrammability makes these chip sets the most flexible of all the microprocessors.

Table 2 summarizes the progression in IC technology. It should be noted, however, that these technologies are not confined to microprocessors. Any technology available for microprocessor fabrication (LSI) can naturally be used for custom LSI or even MSI to implement anything.

Minicomputers

Minicomputers may have the widest range of performance parameters of any class of uniprocessors. At the low end, minis are often outperformed by high end micros. At the high end, a mini overlaps medium-scale

Table 2. Microprocessor Fabrication Processes

	TECHNOLOGY	CYCLE TIME (μsec)	SPEED-POWER PRODUCT (pJ)	PACKING DENSITY (gates/mm^2)
1972	PMOS	20	40	30
1973	NMOS	2	10	100
1974	TTL	0.2	100	10
1975	IIL	0.2	0.5	100
1977	ECL	0.06		
1980(?)	JTL*	0.001	0.005	

Each year has brought substantial improvements in circuitry capabilities.

*Josephson Tunneling Logic

computer performance (e.g., PDP-11/60, PDP-11/70). Also, just as microprocessors/microcomputers (there is a difference) come either microprogrammed (i.e., user has a fixed instruction set), microprogrammable (user makes up firmware to emulate target instruction set) or hard-wired (special purpose), so do minicomputers. Most newer minis have microprogrammed control store and some have a portion of the control store set aside for user customization of the instruction set (e.g., AN/UYK-20, Interdata 8/32, Varian 73, etc.).

Instruction execution times vary from less than 300 nanoseconds to over one microsecond. Memory addressing schemes include memory management, virtual memory, and paging, among others. Physical memory can range from as little as 8K to as many as a million words. Optional features abound, often including exotic memory schemes, variations on I/0, DMA and interrupts. Floating point arithmetic is becoming popular as either standard at the upper end of the lines or an add-on option (effectively a peripheral) at the mid or lower end. Hardware multiply and divide are often available, whereas current microprocessors (with perhaps one or two exceptions) do not supply this function.

The minicomputer family is by far the largest and most widely used group of SISD machines currently on the market. A mini may be relegated to such relatively menial tasks as supporting a small service bureau, or it may be part of a complex real-time processing or process control system. Groups of minis are often used in networks (such as C.mmp at Carnegie-Mellon University) to achieve, through distributed processing, the power of a single "supercomputer" at a fraction of the cost and at a much greater

level of availability. Such applications address difficult problems, such as speech recognition, nuclear reaction calculations, etc.

Mil Spec Computers--Generally, military computers (e.g., AN/UYK-20, AN/AYK-14, AN/UYK-16, AN/UYK-30, etc.) are minicomputers which have been repackaged to meet military environmental requirements. (For instance, there is no difference, architecturally, between a Raytheon 704 and an AN/UYK-16.)

The specifications imposed by the military on computer manufacturers are generally mechanical and electrical rather than architectural. Certain I/0 protocols may be specified (e.g., NTDS for UYK-20), but in general, the emphasis is on reliability, physical size, weight, and power consumption, as well as other environmental considerations, such as shock and vibration, temperature, EMI shielding, and radiation.

Medium-Scale Computers

Medium-scale computers fill a rapidly dwindling niche between the minicomputer and the large-scale computer. Current top-of-the-line minicomputers are overlapping the medium-scale computers at the lower end, and many of the large-scale computers are more cost-effective than the top end medium-scale computer. An archetypical medium-scale computer might be the very successful Burroughs B3700.

Large-Scale Computers

The patriarchs of the SISD family are the large-scale computers, the DEC SYSTEM 10, Burroughs B6700 and up. These types of machines have a basic clock of 8 to 20 megacycles, and one- or two-million-word (48 to 64 bit) memories, usually backed up by a large amount of disk and tape memory. They have large very complex master control programs and are usually made to accept higher level languages.

An exhaustive poll or study of the market will indicate that there is no standard by which to measure large scale CPU's. Comparisons may be made between models from the same manufacturer. Major mainframe manufacturers often compare their product against specific models of a competitor's line. Generally, each manufacturer is strong somewhere where most of the competition is weaker, but invariably falls short elsewhere.

Several key characteristics seem to be more prone to comparison than most. Since these several characteristics appear foremost in the minds of many users, they may be used to establish what is generally meant by "large-scale computing power."

Basic architecture is not a characteristic of concern here, since on the grand scale, essentially all manufacturers stick to the general von Neumann architecture. The implementation of that general architecture is where one finds differences of significance. The areas within an architecture which appear to be performance adjustors are: 1) main storage;

2) central processing unit registers and data paths; 3) sequence control; and of course, 4) input/output. As was stated earlier, a deficiency in one characteristic is often compensated for by excellence in another.

Main storage is the area which most affects the other performance adjustments. It is in itself adjustable in several ways, each with major and subtle effects elsewhere.

The speed of main storage is the most obvious characteristic or variable to consider when looking at a machine. Typical main memory for a large processing system currently in common use is core. Read-write core typically runs from a 2.5 microsecond cycle time down to much less than 1 microsecond. There are faster storage devices of course (e.g., solid-state memory), but they are not currently typically found as main memory in many large existing installations.

Word size is also critical when classifying a computing machine: the more information a word contains, the more data the machine makes available to process per fetch, and the faster it can store away results. Word size ranges typically, in large machines, between 32 and 64 bits.

More subtle are the ramifications of sheer volume of high-speed, first-level memory. The more high-speed local memory available, the fewer accesses the processor must make to other slower external storage media.

The second performance parameter (CPU) also has several subfactors which determine its general characteristics. Most obvious of these factors is speed. The fundamental determinant of CPU

speed is the time required to take data from the internal registers, process the data, and get the result to a register. This is determined by a number of factors, such as circuit family, location of general floating registers, as well as the width and speed of these registers. CPU cycle time for a large machine is typically between 1 microsecond (for older machines) and 20 nanoseconds.

Sequence control, like the CPU, has a cycle time. It too is dependent upon several variables, including circuit family. However, sequence control speed is also dependent upon the extent to which microprogrammed sequence control is used relative to sequential logic. Once again, a 25-to-1 ratio can be seen. An older, slower, large machine may take up to 1 microsecond to execute a basic operation, where a faster machine with little or no microprogrammed instruction may run as fast as 20 nanoseconds. However, the cost penalty paid for the speed may be substantial. The recurring cost of the fast sequential logic, as well as complexity of design and maintenance, may be greater than the 25-to-1 speed-up it produces.

Input/output control or processing is an outstanding characteristic or parameter determining machine throughput. In most cases, a manufacturer builds a general set of peripheral devices with controllers which he will use with machines having a wide range of performance. The difference in I/0 rate between machines occurs for two reasons. The most obvious is bus width between processor and peripheral device. The most important difference, however, is the amount of hardware devoted to I/0 tasks. In a basic machine, the CPU communicates directly with the peripherals and must stop all other processing to do so. If this were the

case for a large machine, given the other previously established criteria, we would call the machine "I/0-bound." At the opposite end of the spectrum, the very large-scale machines such as the B7700 essentially free the CPU from doing I/0 by having one or more special purpose I/0 processor. Again, a 5-to-1 ratio, 2 million bits per second, and up to 10 million or more bits per second, are commonly attained.

Large-scale computers (SISD) are used for a variety of tasks but not nearly as commonly as the minicomputer. No doubt, the largest user of large-scale computers is the U. S. Government. Banks and insurance companies, as well as most big business organizations, employ large-scale computers and sometimes networks of large scale computers to keep records, control complex plant automation, forecast business trends, etc. But for the extremely large scientific applications, large-scale, general-purpose SISD machines, even in multiprocessor configurations, are inadequate.

CONCURRENT PROCESSING ARCHITECTURES

The Spectrum of Distribution

Distributed processing means many things to many people, unfortunately. It is still in 1977 a fairly esoteric discipline with little, if any, standards, although the Distributed Processing workshop held at Brown University in August 1976 and August 1977 and a wealth of other gatherings of top scientists involved in distributed processing are attempting to codify this nebulous area. [CAN 76]

Distribution may occur at many levels. One can visualize a spectrum with the ARPANET at one extreme and a processor such as the RAM-1 at the other. [RAM 76]

ARPANET by some definitions is not distributed processing at all, but by others it may be considered a very loosely coupled, geographically distributed, heterogeneous network. In some sense then, it is distributed processing where there is no unit of cooperation but simply communication between the nodes in the network. [BELL 74]

Close to the opposite end of the spectrum is C.mmp--a very tightly coupled homogeneous network using parallel processing to achieve its throughput objectives. C.mmp divides single programs into pieces and parses the pieces out to several cooperating parallel processors (PDP-11's). [BELL 72], [WULF 75]

Taking this concept one step further, processors like the RAM-1, designed at Brown University and the Moore School of Electrical Engineering at the University of Pennsylvania, will probably, by 1983, be as commonly used as today's PDP-11's. The RAM-1 uses distributed processing as an architectural tool--distributing at the lowest possible (i.e., fixed function) level where the distribution problems are hidden from the user. Through additional architectural techniques, a fairly high-performance machine can be realized out of a network of bit-slice

microprocessors. So the unit of cooperation in C.mmp is a phase of a program, and the unit of cooperation* in a RAM-1 type configuration is the individual instruction.

Functional distribution is of greatest interest and common usage now. As was pointed out above, functional distribution may occur at various levels. Furthermore, a distributed processor may be general purpose or special purpose, and in either case, the individual processors which make up the processor may be very much heterogeneous or homogeneous.

It is worthwhile to note that although essentially all well-known concurrent and/or distributed processing systems achieve or surpass large-scale SISD performance, it is not the case that these classes of architectures (SIMD, MISD, MIMD) are necessarily high-performance machines. That is, it is by design and not *generic definition* that the machines commonly thought of as concurrent and/or distributed processors generally outperform the SISD machines.

SIMD Machines

In the early 1960's, when it became apparent that the scientific community was beginning to formulate problems too large and complex for large SISD machines and even large-scale multiprocessor machines, the SIMD

*The term "unit of cooperation" as used here is related to the term "homogeneous problem" as referred to in C. Vick's definition of distributed processing.

architectures began to take more form. SIMD was not a radical new idea in the 1960's, but the thought of actually implementing a fast giant like ILLIAC IV seemed like science fiction. Most of the problems of implementing and operating ILLIAC were based, again, on semiconductor manufacturers' problems. ILLIAC could probably be rebuilt today with current LSI technology and would result in a much more effective, reliable machine. (Of course, there would still be the problem of figuring out how to use it.)

The SIMD machine organization has N processors controlled by a single controlling processor. These N processing elements will work in parallel, all (or a selected subset) performing the same operation simultaneously on different data. All nonselected processors do nothing. They may share the same single memory or a set of N independent memories, the memories together making up the main memory for the computer.

With N processors, it is better to have N independent memories because each processor can have immediate access to a location for storage of frequently used programs and results. Having N memories resolves some of the conflicts created by N processors requiring memory access to a single memory. The memories could be connected through a cross-bar, but the delay, the complexity, and the cost for conflict resolution for such a system would be counterproductive for large N (since the number of active elements and links increases as N^2). [SULL 77] Because of this, in most parallel computers, each memory unit is associated with one of the processing elements, and only that processing

element has immediate access to the memory. Therefore, there must be interprocessor communication to get a memory reference from one memory element to a processing element not associated with that memory element.

Processor connections are needed for such interprocessor communication. The kind of connection used affects the way the computer solves certain problems. Some connection schemes are better suited for solving one type of problem than another.

One interconnection pattern known to be good for a broad class of problems is the cyclic shift used in ring networks (e.g., Farber's DCS [Farb 72]). This gives each processor two neighbors. The path of connections goes around a circle. With unidirectional paths, the maximum number of routing steps required to pass information from one processor to another is N-1. With bidirectional paths, the worst case is (N/2)-1. For a number of parallel algorithms, (e.g., Fourier transforms, sorting, and matrix transposition) the ideal connection is a "perfect shuffle" connection. The information paths between processors follow the pattern that cards fall in when they are perfectly shuffled. If a deck of cards falls in when it is perfectly shuffled, the cards would be divided into two equal piles and then intermingled so that the first card of the first half was first, the first card of the second half was second, the second card of the first half was third, etc. The perfect shuffle connection is not optimal for several other applications. Because solving partial differential equation requires iterations that effectively update a grid point as a function of the values of nearby grid points, Slotnick proposed the array structure of interconnection now used in the ILLIAC IV. Each processor has four neighbors, and the paths are bidirectional.

A complete interconnection network, where each processor is directly connected to every other processor, is the most flexible connection scheme, but the cost required for N processors increases proportionally with the square of N (as stated before). Thus for large N, this is not a realistic connection scheme. The best design would emphasize connections most widely used in SIMD algorithms in comparison with SISD algorithms.

As a rule, the SIMD machines do not require multiprogramming techniques to achieve high throughput.

Quite often, associative techniques, such as those used in PEPE (to be explained later), are employed. The STARAN is another example of an associative SIMD machine--the only one commercially available. [STAR 75]

Fault tolerance can be incorporated into this architecture fairly easily. When an element or elements fail, their load may be picked up by others until repairs can be made. By distributing the processing and the control, single failures are usually not catastrophic. A good distribution will allow a good MTBF and a minimal MTTR, resulting in a much more reliable, useful system.

ILLIAC IV--ILLIAC IV is a SIMD Array Processor. It fully implemented today, it would be the world's largest processor. Unfortunately only one-fourth of ILLIAC's processing elements were built. This was due to both the inability of certain semiconductor manufacturers to keep promises and also due to software limitations.

The processing elements are grouped into 8 x 8 arrays; each array is called a quadrant. There is a single control unit for each quadrant, and each array-control unit set was to have communicated with the other three in a ring. Within the 8 x 8 array, each individual processing element (PE) is connected to its four nearest neighbors. A processing element is essentially a general-purpose computer with most of the control outboard (i.e., in the control unit for the array). Since all PE's execute their instructions in parallel, most control signals are generated remotely in the control units (CU), thereby considerably reducing PE implementation costs. Figure 4 shows ILLIAC as it was supposed to be built, and Figure 5 illustrates the PE and CU structure.

The ILLIAC IV has almost no operating system. A user takes hold of the system, runs his problem, and then lets it go; the next user does the same thing. In the smaller computers (PDP-11s) that surround ILLIAC's control unit and processing elements, there is more complex software that attracts a queue of users waiting to get at the big machine and allows them to perform nonarithmetic companion processes. But the actual ILLIAC operating software itself is very simple.

ILLIAC is used for a variety of problems for which virtually no other machine is suitable today. The NASA Ames users, who now take up about 20 percent of ILLIAC IV's operating time, are solving two-dimensional aerodynamic flow equations. (These involve the solution of complicated partial differential equations.) The remaining users are running weather prediction and climate models, and several types of signal processing

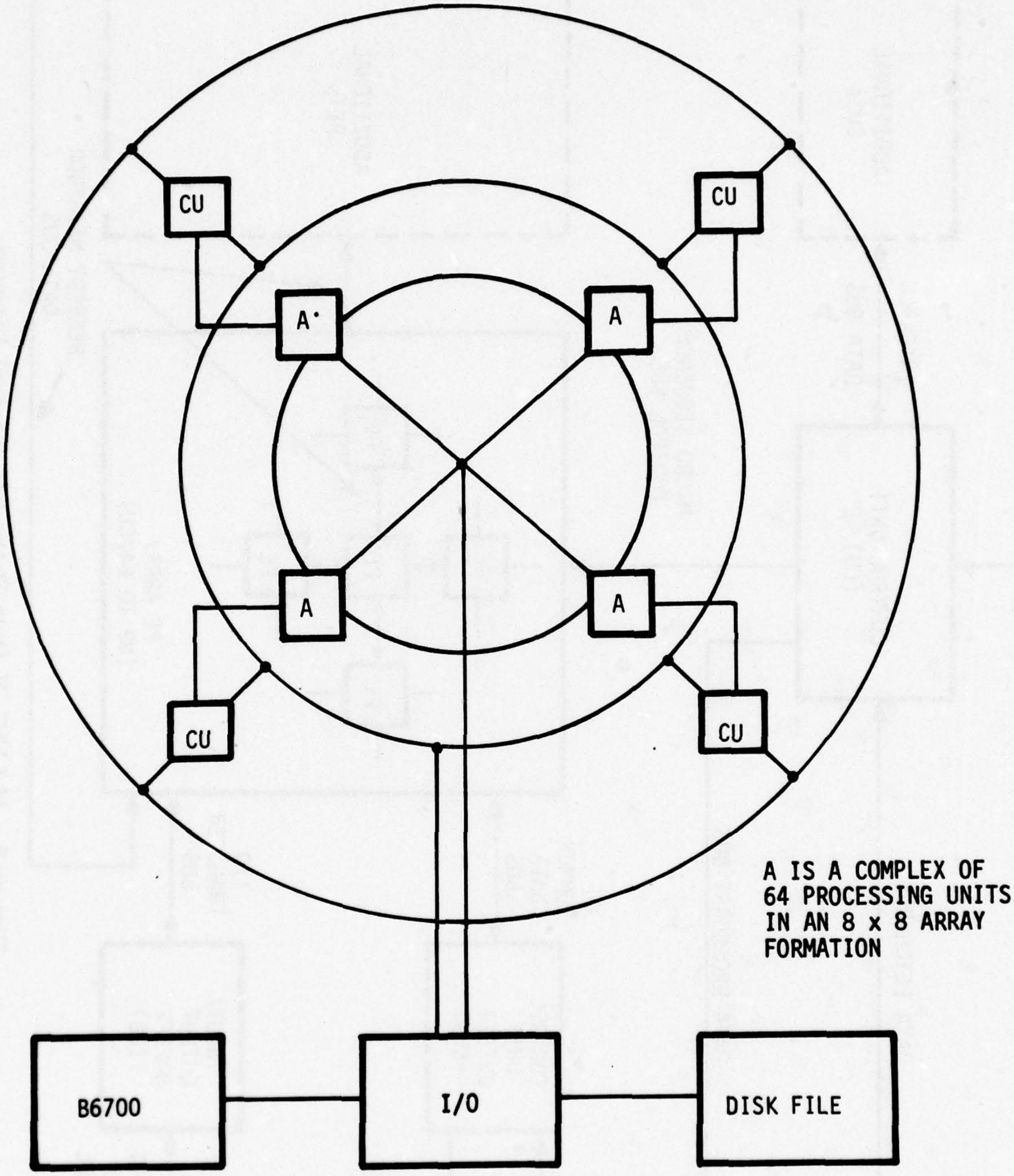


Figure 4. ILLIAC IV System

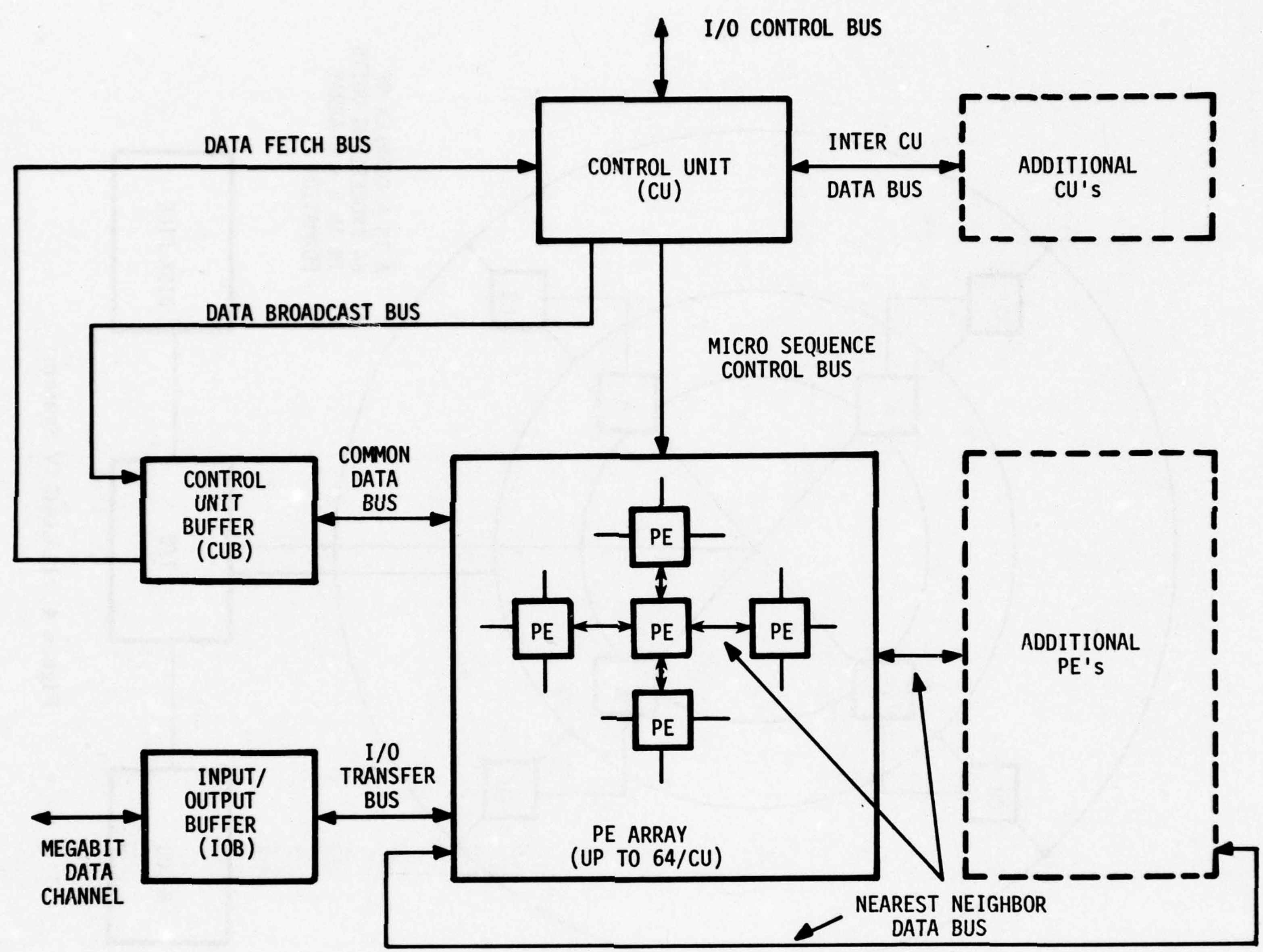


Figure 5. ILLIAC IV Data Paths and Control Linkage Among System Elements

computations, including fast fourier transforms. Other applications include solving problems involved in beam forming and convolution, seismic research, and radiation transport for fission reactors.

The original design of ILLIAC IV with its four arrays was supposed to perform 1000 million floating point operations ("megaflops") per second. Expectations dropped when the machine was limited to one processor array, to about 200 megaflops. This goal has not been realized yet, though ILLIAC IV does currently achieve 15 megaflops. The CDC 7600 at Ames has a theoretical limit of 10 megaflops but has only been run at 5 megaflops.

PEPE--PEPE is an associatively addressed ensemble of parallel processing elements. [CRAN72] It uses associative access methods to store or retrieve data from a processing element. There are no direct communication paths between elements, as was the case in ILLIAC IV, except through the control unit; for this reason, PEPE is termed an ensemble. Because the data base that PEPE works on represents objects that interact very little with each other, there is no need for direct intercommunication.

Figure 6 shows a very high level diagram of PEPE, whereas Figure 7 shows a next lower level view of a processing element (PE). The final PEPE design has 288 elements, but the currently implemented system has only 11 elements.

PEPE is a parallel processor designed to perform real-time radar tracking. It was designed specifically to augment (offload) a general-purpose machine (e.g., CDC 7600) in a ballistic missile defense environment. The application speed requirements dictate the ability to input,

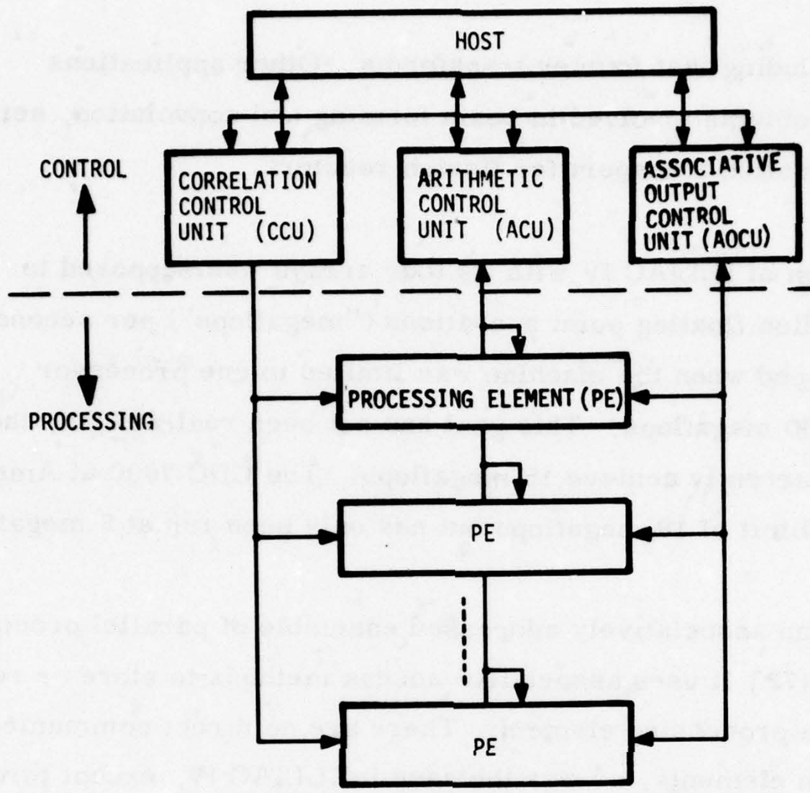


Figure 6. PEPE Block Diagram

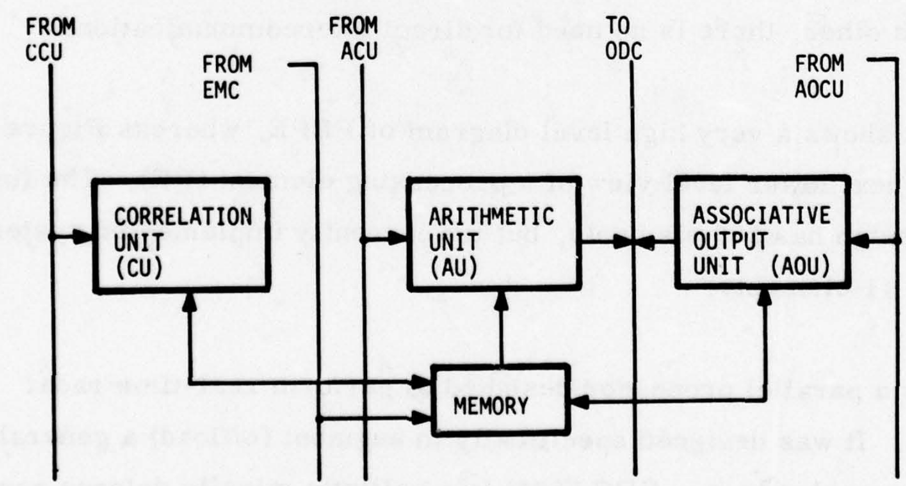


Figure 7. PE Block Diagram

process, and output data simultaneously. This requirement is met by simultaneous operations in each PE. PEPE can be inputting data through the Correlation Unit/ Correlation Control Unit Complex (CU/CCU), computing on tracks in the Arithmetic Unit/ Arithmetic Control Unit Complex (AU/ACU), and outputting through the Associative Output Unit/ Associative Output Control Unit Complex (AOU/AOCU), as seen in Figure 8.

The PEPE prototype has 16 processing elements arranged in two wings of eight processing elements each. Each processing element consists of an arithmetic unit, a correlation unit, an associative output unit, and a memory. A global control unit, consisting of an arithmetic control unit, a correlation control unit, and an associative output control unit, serves to couple the ensemble of processing elements to the host. There is no interprocessor communication.

The processing elements perform arithmetic processes, such as track updating and track prediction. Only "active" processing elements will perform the current instruction broadcast by the controller, and all the inactive processing elements will remain idle. Each processing element contains an 8-bit, content-addressable tag register together with the contents of the processing element. Using this ability, all object ranges can be simultaneously searched to find maximum, minimum, and inbetween limits. These processes can be combined to perform ordered retrieval.

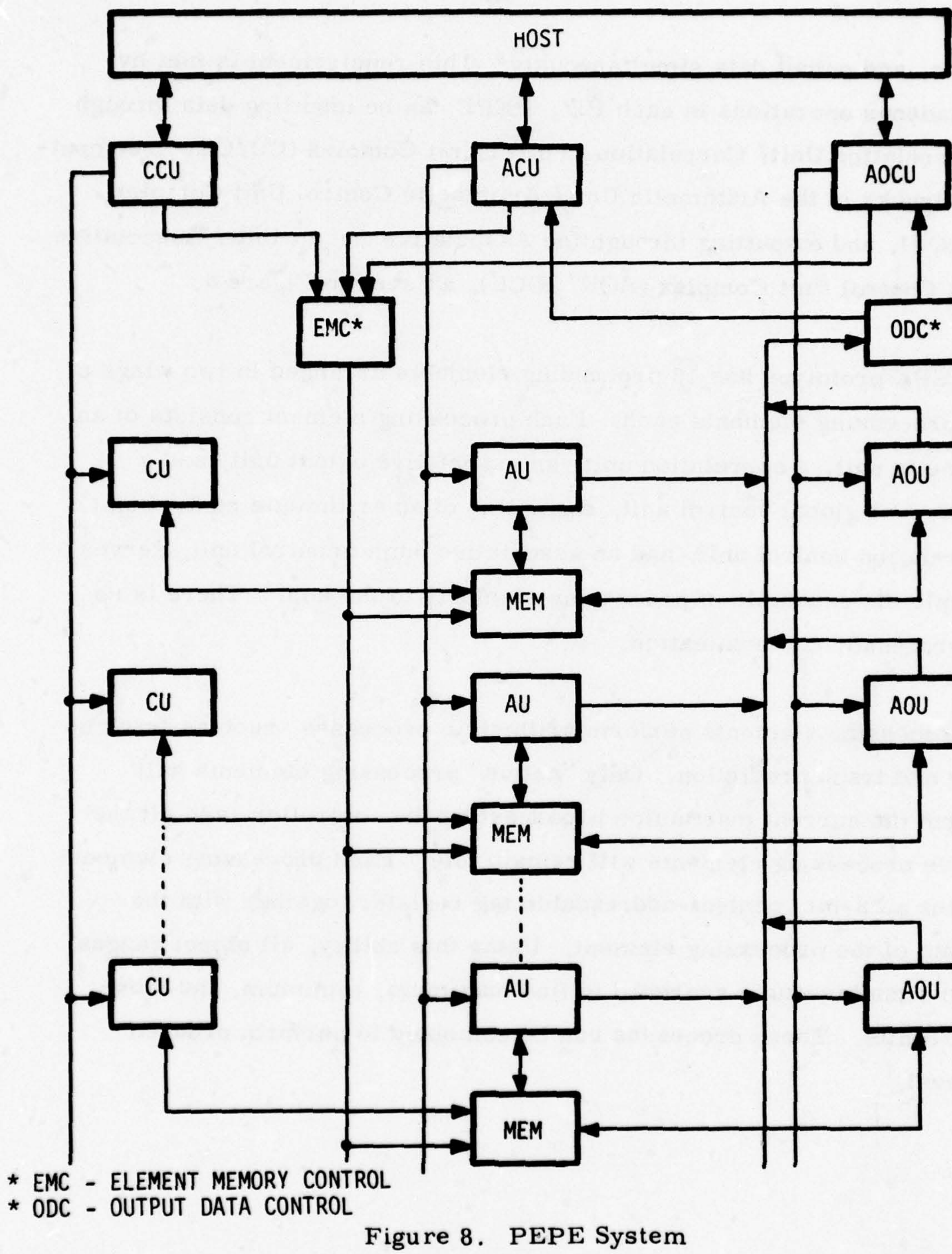


Figure 8. PEPE System

The associative output unit and the associative output control unit were added to the initial design in order to meet I/0 considerations not initially anticipated. The operation is similar to that of the arithmetic units, but for the lack of arithmetic ability. Its function is data sorting, and it is the primary means of system output.

The correlation unit is essentially a content-addressable memory. It has eight words of 40 bits each in each unit. Used for associative comparisons of incoming data, it is only able to handle integer operations. Its instruction repertoire consists primarily of associative memory match and store instructions.

The PEPE can best be described as an associatively organized, highly parallel computer. It is a SIMD machine like the ILLIAC IV, but unlike the ILLIAC IV, it does not have direct processor interaction.

An operation on a pair of operands will take longer on a content addressable parallel process than on a conventional machine, because it will be serial by bit rather than parallel by bit. But if the same operation is to be done on N pairs of operands, the conventional machine will take N times as long as it does for one pair where the time for the content addressable parallel processor will be the same as for one pair of operands.

On a conventional machine, finding the largest element of an array requires time of order N, while in the content addressable parallel processor it requires time of order one (independent of N). The time required to compare N elements with each other is of the order of N squared in a conventional machine while in the content addressable parallel

processor it is on the order of N. In general, for most problems it is fair to say that content addressable parallel processors reduce the complexity of the problem by an order of N. [FOST 76]

SIMD Drawbacks--The basic constraint on a SIMD machine is that each memory module can satisfy only one request per memory cycle. For best operation, the N operands of a vector operation will lie in distinct memory modules. This presents an interesting problem when storing a matrix in a SIMD computer made with the processing elements connected in an array structure. It is clear that for a matrix to be operated on row-by-row, the most efficient storage of that matrix would associate each element of each row with a different processing element. If a matrix is to be operated on column-wise, it is necessary to have each column element in a separate processor. For this reason, matrices in SIMD machines are stored in a storage format known as skewed storage. Successive rows are cyclically shifted by unit amounts so each column and each row is distributed among several memory elements. With skewed storage, both the rows and the columns of the matrix can be addressed in unit time.

SIMD architectures are typically very compute-bound, special-purpose machines. They are best suited for problems using iterative algorithms (e.g., solving boundary value problems) typical to weather bureau, nuclear reactor, and target tracking signal processing applications. They are not useful for general applications because of the difficulty in finding parallel algorithms suitable for an array processor. A great deal of this problem lies in the years the industry has spent designing serial algorithms, while ignoring parallel algorithms.

A large number of parallel machines have been proposed over the last 20 years, but only three designs have been built: ILLIAC IV, PEPE, and OMEN by Sanders Associates. The first two machines have been described.

MISD Machines

MISD machines are commonly called pipeline machines. Pipelined processors are well suited for processing vectors and arrays. In fact, many of the large machines using extensive pipelining also feature vector instruction capability. Peripheral processors are also available for use with a host general-purpose computer to give vector processing capabilities, as well as microprogrammed routines for functions useful in signal processing applications (e.g., the IBM 3838, a stored program computer usually used as an attachment with a System 370).

Traditional computers execute scalar instructions with two source and one destination operands. A machine with a vector instruction set operates on operands which are vectors and may contain hundreds or thousands of elements. In a vector instruction, each word of the vector will undergo the same operation. Vector instructions may operate on two-vector operands or on a one-vector operand (e.g., sum the elements of a vector). Vector operations, when implemented on a scalar machine (i.e., SISD), will typically take the form of a loop such as:

```
DO 10 I = 1,500
10 A (I) = B(I) + C(I)
```

This can be reduced to a single statement on a vector machine.

A vector machine may be considered a special case of a pipeline or vice-versa. The CDC 6600 and CDC 7600 utilize intermediate storage for all operands, only allowing one instruction to use a functional unit at a time. Their architectures and applications led to the CDC STAR, which streams operands and the CRAY-1, which chains operations. These machines are important in that they can be used on scientific problems in which the scalar machines would be completely bogged down. But the problem is that most machines specialized for vector operations are not necessarily good at scalar operations.

A recent study done at Los Alamos Scientific Laboratories [RUDS77] has determined that unless a vector machine in scalar mode can at least compete with the scalar machines, they are not worthy of consideration for anything but highly specialized applications. Rudsinski and Worlton state that:

"Figure 9 shows the performance bounds for three different ratios of scalar performance in the machines being compared. The values on the ordinate are the ratio of the performance of the scalar-vector machine to the existing scalar machine. The first scale indicates that both machines perform scalar operations equally well. Note that even for an infinitely fast vector processor the performance gain from 50 percent vector work is at most a facto' of two; the performance gain from 75 percent vector work is at most a factor of four; etc. This limitation on the performance gain is entirely due to the scalar work that remains after the time to complete the vector work is set to zero.

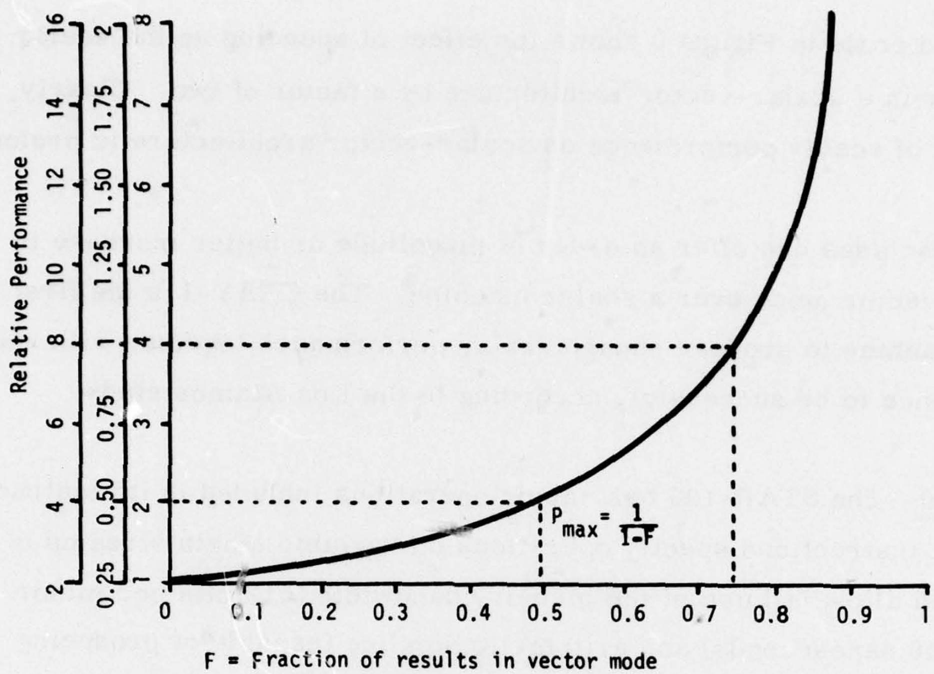


Figure 9. Maximum Performance Gain from Vectorization

"The second scale in Figure 9 illustrates the situation where the scalar speed of the scalar-vector computer is significantly slower than that of the existing machine. In particular, this ordinate displays the case where the scalar mode of the new machine is one-fourth that of the existing machine. Note that we must have 75 percent vector results just to "catch up" with the performance of the existing scalar machine. Because we are also assuming infinite vector speed, in a real machine with these scalar characteristics, vector results will have to be somewhat higher than 75 percent to "break even." This illustrates the severe penalty of incorporating a slow scalar processor into a scalar-vector architecture.

"The third scale in Figure 9 shows the effect of speeding up the scalar processor in a scalar-vector architecture by a factor of two. Clearly, the effect of scalar performance on scalar-vector architecture is profound."

Vector machines can offer an order of magnitude or better increase in speed in vector mode over a scalar machine. The CRAY-1 is the first vector machine to produce enough scalar performance together with vector performance to be successful, according to the Los Alamos study.

STAR-100--The STAR-100 has vector operations included in its instruction set. The instructions specify operations on variable length streams of data which allow full use of the memory bandwidth (512 bits per minor cycle of 40 nanoseconds) and arithmetic pipeline (capable of producing 100 million 32-bit, of 50 million 64-bit, floating point results per second).

The heart of the central processor is the stream control unit. The stream control unit includes an instruction control unit that has a stack of 64 instructions filled via a hardware look-ahead of eight to sixteen instructions, i.e., it can prefetch a block of instructions while processing takes place from the stack.

The STAR-100 stream unit also has a 256-word register file, addressed as the lowest part of main memory, for temporary sources and destinations for register operations. The stream control unit controls the arithmetic unit and the string unit. The floating point arithmetic unit is divided into two units which can operate in parallel: one pipeline for add, subtract, multiply, and similar vector instructions; and a second pipeline for divide, square

root, and vector instructions. The string unit is for operating on strings of binary and decimal numbers, i.e., variable length operands such as those required for character string operations. All pipelines have a minor cycle time of 40 nanoseconds.

Three channels from the memory to the arithmetic unit allow two vectors to be inserted into the top of the pipeline and a result stream to be sent back to memory, all simultaneously. An inhibit bit stream may also be used to inhibit the designated operation on particular elements of a vector.

The STAR-100 has approximately 130 scalar and 65 vector instructions. It is currently used at Lawrence Livermore Radiation Laboratories. Like many of the supercomputers, it was several years late in its delivery.

TEXAS INSTRUMENTS-ASC--The TI-ASC central processor is built around an instruction processing unit (IPU). This preprocesses the instruction stream, which typically contains a mixture of scalar and vector instructions. The IPU is pipelined, with four segments to the pipeline. These are controlled by the IPU.

The central processor may have one, two, three, or four pipelined execution units. Each unit can have up to eight instructions in different stages of execution in the pipeline.

The central processor has sixteen base address registers, sixteen arithmetic registers, eight index registers and eight vector parameter registers. These last registers are used to extend the instruction format for the com-

plete specifications of vector instructions. They are the vector control registers which contain the parameters (vector control blocks) that describe vectors to be operated on. The vector parameters can provide three dimensions of addressability. This corresponds to the nesting of three index loops in FORTRAN for a scalar operation machine and can be used to process a three-dimensional array.

The vector capabilities of the control processor are made available through the use of VECTL (vector operation after loading the vector parameters) and VECT (which assumes the parameters are already loaded) instructions. The TI-ASC has 177 scalar and 70 vector operations that it can perform.

The processor has operand look-ahead capability, which requests memory references prior to the time of actual need. Each arithmetic pipeline has a memory buffer unit that provides smooth data flow to and from each unit. The X and Y operand streams and the Z result stream are all double-buffered. A TI-ASC buffer is an eight-word set of data, used because the TI-ASC fetches and stores data in groups of eight words, so that each stream has a sixteen-word buffer associated with it.

The TI-ASC, as all vector machines, reaches its maximum throughput when in vector mode. Each pipelined arithmetic unit can produce a result every 60 nanoseconds. A processor with four pipelined arithmetic units can produce four results each 60 nanoseconds, or an effective rate of one result every 15 nanoseconds.

Because the TI-ASC can be configured with multiple pipelines, the TI-ASC compiler software is responsible both for the generation of vector instructions and the partitioning of vector operations over multiple pipelines. For mixtures of vector and scalar instructions, the compiler enforces proper precedence through the use of directives for the central processor to operate in parallel mode (FORK) or sequential mode (JOIN). Programs compiled for a one-pipeline ASC will execute correctly on a multiple pipeline system, but performance will be increased via recompilation for the multiple pipeline machine.

There are several TI-ASC machines in use today. The first model is now being used for software development. The second model is now in Amstelveen, Holland, devoted to seismic data processing work. Seismic operation is characterized by large data bases and many job steps of long computational sequences. The third model is being used by the U. S. Army for ballistic missile defense. The fourth model is used by the National Oceanic and Atmospheric Administration. Preliminary reports of the performance of this last system on a weather benchmark indicate it is several times faster than any conventional computer systems; it is about five times faster than a CDC 7600 or an IBM 360 model 195, and about forty times faster than an IBM 360 model 65.

CRAY-1

The computation section of the CRAY-1 consists of an instruction control unit, operating registers, and functional units. The instruction control performs all decisions related to instruction issue, and coordinates the

three types of processing: vector, scalar, and address. Associated with each type of processing are registers and functional units for that processing mode. For vector processing there are:

(1) a set of 64-bit, 64-element vector registers;

(2) three functional units dedicated to vector applications: add, logical, and shift; and

(3) three floating point functional units to support both scalar and vector operation: add, multiply, and reciprocal approximation, the latter being used for division.

The flow of data is generally from memory to registers, then registers to functional units. The flow of results is from functional units to register, and then from register to memory, or back to functional unit. Data flows either on a scalar or a vector path, the one exception being scalar registers, which can be used as one of the operands required for vector operations (e.g., adding a constant to each element of a vector).

In contrast to the TI-ASC and STAR-100 in which scalars and vectors share operand and result registers, and all operations share the same arithmetic units, the CRAY-1 has vector functional units that operate solely on vectors and eight vector registers dedicated only to vector operands, each with 64 words of 64 bits. The successive elements within a vector register are considered to make up a vector.

Each element of a vector is processed identically. A vector operation begins by taking the first elements of one or more vector registers and delivering the result to the first element of a vector register. Successive elements can be provided each clock period (12.5 nanoseconds). The

vector instruction continues until the number of operations performed equals a count specified by a vector length register.

Results are obtained from the vector functional units every clock pulse of 12.5 nanoseconds. A result will be received by a previously identified register and may be received by a register and transmitted as an operand to a subsequent operation in the same clock period. This allows chaining of vector operations.

The CRAY-1 can achieve parallel vector processing in two modes, either several completely parallel vector operations using different functional units, or by using one vector result stream as the operand stream for another separate functional unit that is operating simultaneously.

There is a vector mask register, 64 bits long, used in conjunction with vector merge and test instructions, allowing operations to be performed on selected vector elements. Each bit of the vector mask corresponds to a word element in a vector register.

The CRAY-1 is the fastest number cruncher available today. It has no assembler or peripherals and depends on a host for operation (although a CRAY-1 processes faster than a CDC-7600 can supply it with information). It has a large, very fast memory (one million words of bipolar memory) with a memory cycle time of 50 nanoseconds and requires 125 nanoseconds to fetch operands from memory to the registers. When operating on vectors, it can supply a register with an operand every 12.5 nanoseconds. The bipolar memory requires that the CRAY-1 be freon cooled.

The CRAY costs close to $10 million, which is considerably more than the $6 million necessary to buy a STAR, which can handle its own peripherals and has a FORTRAN compiler. At best, the CRAY-1 can deliver results twice as fast as the STAR. Grosch's law (that the computing power increases as the square of the cost) seems to break down for very large supercomputers.

Evaluation of performance supplied by Los Alamos Scientific Laboratory shows that the CRAY-1 is from 3.4 to 5.1 times faster than a CDC 7600 (a machine with a theoretical maximum output of ten million results per second) in vector operations on vectors ranging from 20 to 500 words in length. Also, system availability and mean time to failure were found to be well above that of the 7600.

SECTION 4

CONCLUSIONS

In the past, the increase in speed and size of computer systems was due in large part to componentry. There is still close to an order of magnitude increase in speed expected from technological advances in electronics. Other techniques under development will also speed up computers of the future. Super-cooled Josephson junctions are expected to reach the speed of 0.02 nanoseconds. Electro-optical logic elements have already been constructed, and the speed of operation is expected to be below 0.01 nanoseconds. Another potential for speed lies in elastic surface-wave technology, where switching speeds may reach below 0.01 nanoseconds.

Yet the speed of hardware alone will not be able to satisfy all of the demands put on computers in the future. It is clear from the examples presented that architectural design techniques can and will play a large role in determining the performance of computer systems. These techniques will become more prevalent in future machines.

The architectural concepts presented are not limited to large scale, special-purpose architectures. They can be applied to general-purpose machines and can be applied even to small-scale machines. And they can be used without ever having to buy a fancy processor, e.g., the Ramseyer-van Dam multi-processor machine. [RAMS 77]

The explosive growth of LSI and VLSI will have an enormous effect on micro-, mini-, conventional, and supercomputers. Each computer will have its own use, and each application will have a computer suited for it. Many of the techniques discussed will appear in all of these machines. The only difference will be in the amount of cache, parallelism, pipelining, or associative memory present in each machine.

The cost of computers should decrease dramatically with LSI. It is estimated that a processor with the same processing power as the STAR-100 will cost about $50,000 by the middle of the 1980's, a cost reduction by a factor of 120.

Many techniques will be combined. In fact, as the size and cost of components decrease, there will be an increase in the number of machines combining a variety of architectural techniques. For example, most large machines already have some content-addressable memory to handle memory paging. The CDC 6600 and the TI-ASC combine special-purpose processor units with a multiprocessor peripheral control. Distributed processors are being constructed using tightly coupled networks of minis or micros, with collections of processors configured in a pipeline. [FULL 73], [FULL 75], [RAMS 77]

Reliability of components should continue to increase. The availability of computer systems will reflect the increase in the reliability of components; as size and cost decrease, there will be replication of units for the purpose of making computers fault-tolerant.

Complete processors are already available on one chip. When several processors are available on one chip, the creation of array processors and even array processors with backup processing elements will be very easy and inexpensive.

The rapid advances in computing power will continue for some time. And as the alternatives available become greater in number, architectural design techniques will play an increasing role in the development of computer systems. In terms of military computers, the trend will no doubt go towards emulation engines implemented out of microprogrammable microprocessors (e.g., AM2901, M10800). These sorts of engines then may be used in place of current standard mil computers like the AN/UYK-20, but will, in any case, be software-compatible with one or more standard computers at least in one emulation mode.

Much processing currently done by smaller SISD machines will be parsed and distributed to networks of microprocessors and/or microcomputers to achieve more cost-effective, faster, and reliable systems.

Today's general-purpose, large-scale main frames also will eventually follow this basic trend and end up as a set of smaller special-purpose processors which themselves are implemented via a third generation microprocessor or special purpose hard wired processors.

Preceding Page Blank - NOT FILMED

BIBLIOGRAPHY

[ANDE75] Anderson, G. and Jensen, E., "Computer Interconnection Structures: Taxonomy, Characteristics, and Examples," ACM Computing Survey, Vol. 7, No. 4, December 1975.

[BELL71] Bell and Newell, Computer Structures Readings and Examples, McGraw-Hill, 1971.

[BELL72] Bell, C.G., C.mmp: The CMU Multiminiprocessor Computer. Requirements and Overview of the Initial Design, Carnegie-Mellon University, 1972.

[BELL74] Bell, C.G., "More Power by Networking," IEEE Spectrum, February 1974.

[BURK46] Burks, A.W., Goldstine, H.H., and von Nuemann, J., "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," Report to U. S. Army Ordnance Department, 1946.

[CAN76] Computer Architectural News, ACM, Vol. 5, Nos. 5 and 6, 1976.

[CRAN72] Crane, B.A., Gilmartin, M.J., Ruf, P.T., and Shively, R.R., "The PEPE Computer," IEEE Compcon 72 Digest.

[CUSH75] Cushman, R.H., "EDN's Second Annual Microprocessor Directory," EDN, November 20, 1975.

[FARB72] Farber, D. and Larson, K., "The System Architecture of the Distributed Computer System--Communication System." Proceedings Symposium on Computer Networks and Teletraffics, April 1972.

[FLYN66] Flynn, M.J., "Very High-Speed Computing Systems," IEEE Proceedings, Vol. 54, December 1966, pp. 1901-1909.

[FLYN72] Flynn, M.J., "Some Computer Organizations and Their Effectiveness," IEEE Transaction on Computers, Vol. C-21, No. 9, September 1972.

BIBLIOGRAPHY (Continued)

[FOST76] Foster, C.A., Content-Addressable Parallel Processors, von Nostrand Reinhold, 1976.

[FULL73] Fuller, S.H., Siework, D.P., and Swan, R.J., "Computer Modules: An Architecture for Large Digital Modules," Proceedings of the First Annual Symposium on Computer Architecture, University of Florida, Gainesville, 1973.

[FULL76] Fuller, S., Swan, R.H., and Sieworek, D.P., "The Structure and Architecture of Cm*, Preliminary Paper for Workshop on Distributed Processing, Brown University, August 1976.

[OGDE74] Ogden, J.L., "Microprocessors: Promises and Practices," Microprocessor Architectures, IEEE Intercon Technical Papers, 1974.

[RAMS77] Ramseyer, R.R. and van Dam, A., "A Multi-Microprocessor Implementation of a General Purpose Pipeline Machine," Proceedings of the Fourth Annual Symposium on Computer Architecture, March 1977.

[RUDS77] Rudsinski, L.E. and Woreton, W.J., "The Impact of Scalar Performance on Vector and Parallel Processors," NCC '77.

[SLOT62] Slotnick, D.L., "The Solomon Computer, FJCC, 1962.

[SLOT67] Slotnick, D.L., "Unconventional Systems," SJCC, 1967.

[STAR75] "STARAN System Description," Akron, Ohio: Goodyear Aerospace Corporation, 1975.

[SULL77] Sullivan, H. and Bashkow, T.R., "A Large-Scale Homogeneous, Fully Distributed Parallel Machine, I." Proceedings of the Fourth Annual Symposium on Computer Architecture, March 1977.

[SWAL74] Swales, N.P., and Weisbecker, J.A., "COSMAC-A Microprocessor for Minimum Cost Systems," Microprocessor Architecture, IEEE Intercon Technical Papers, 1974.

BIBLIOGRAPHY (Concluded)

[WILK51] Wilks, M.V., "The Best Way to Design an Automatic Calculating Machine," Manchester University Computer Inaugural Conference, July 1951.

[WULF75] Wulf and Levin, "A Local Network," Datamation, February 1975, pp. 47-51.